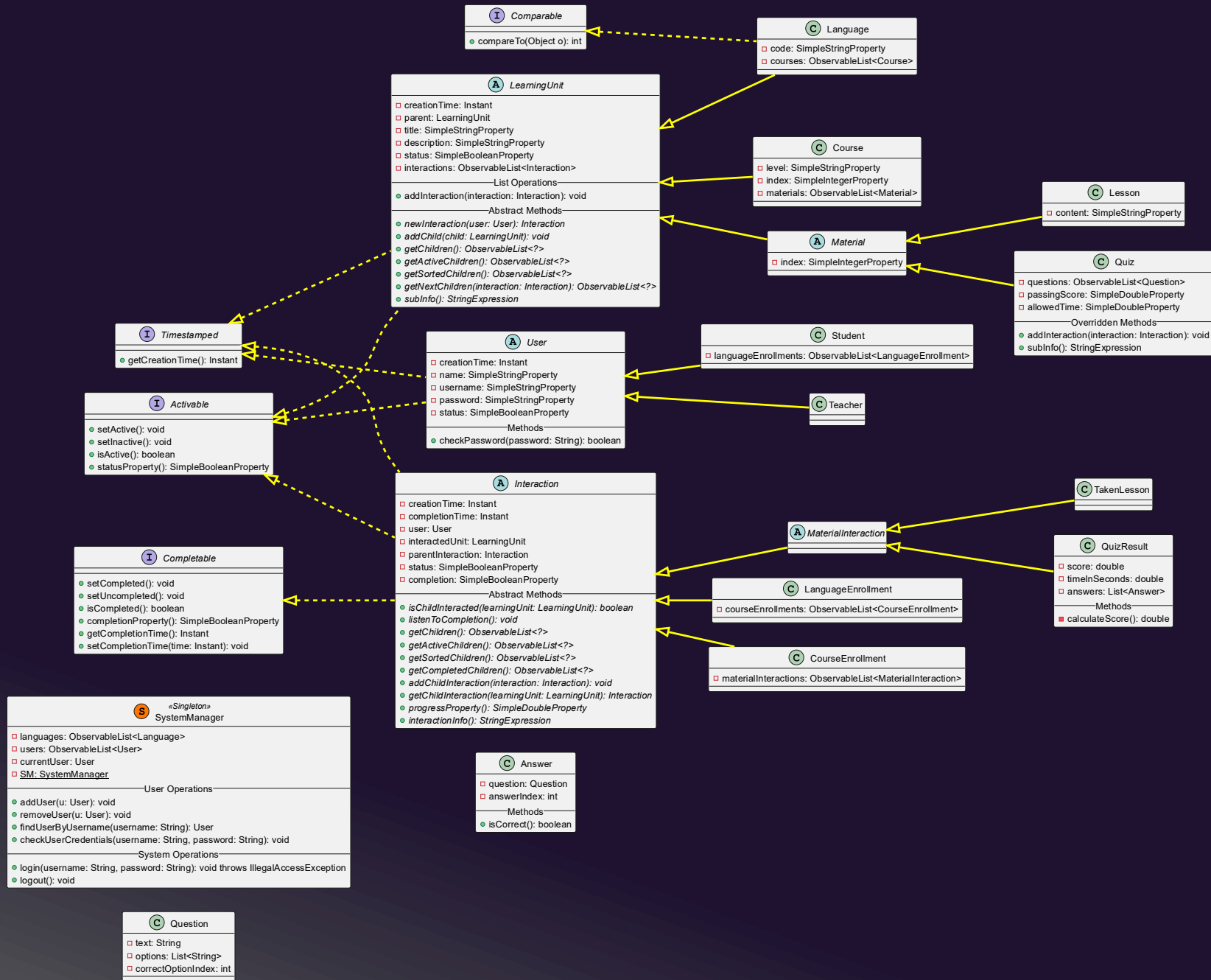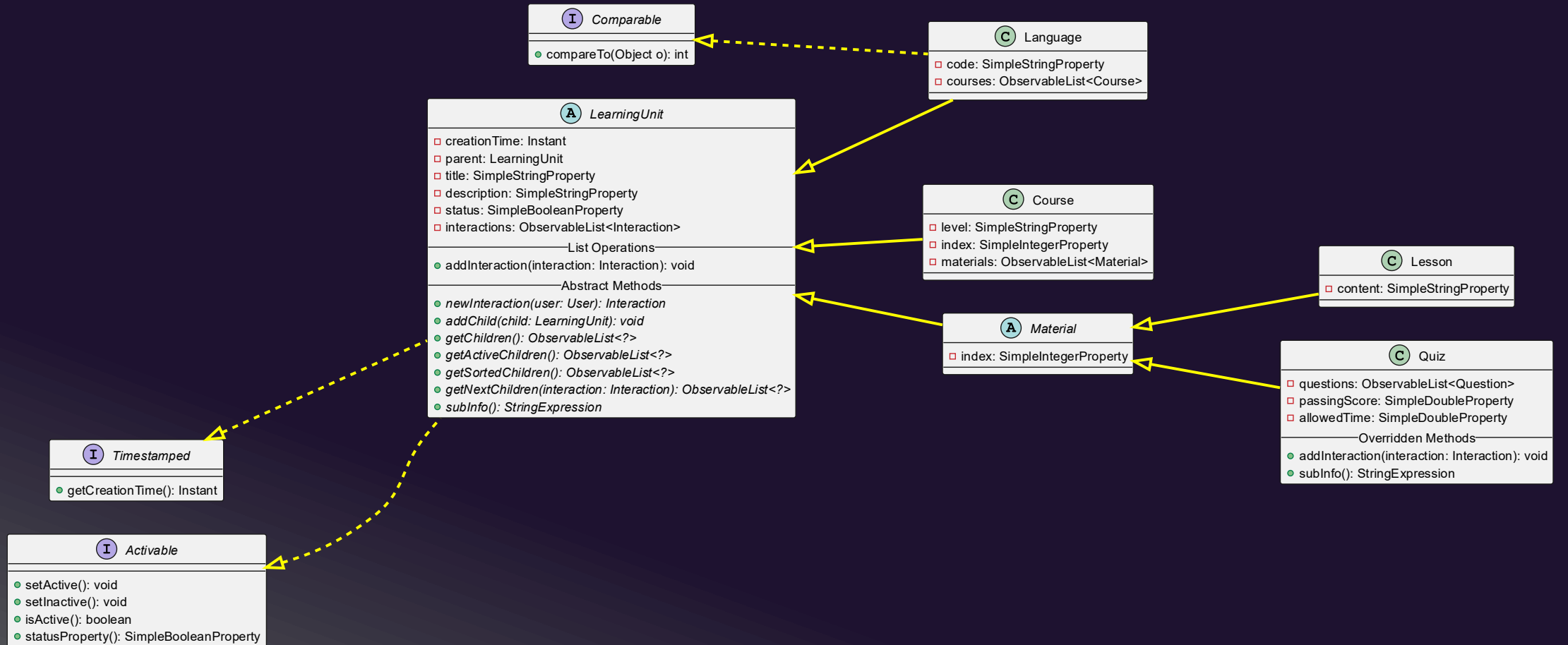# Online Language Teaching System

CSE231s Advanced Computer Programming
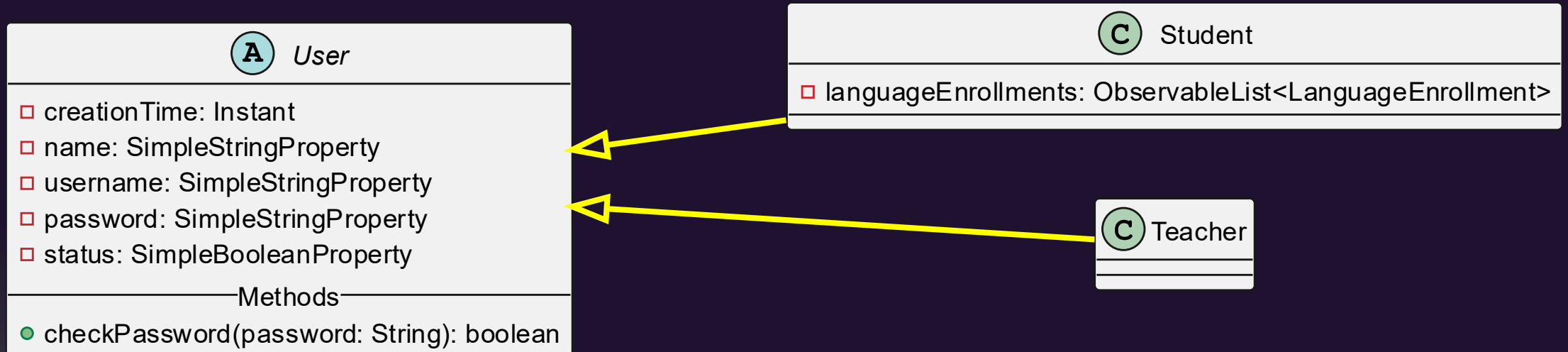
```java
List<Engineer> teamMembers = Arrays.asList(
        Omar_Mohamed_Hassan,
        Ammar_Mohamed_Abdelghany,
        Mohamed_Elsayed_Moustafa
);
```
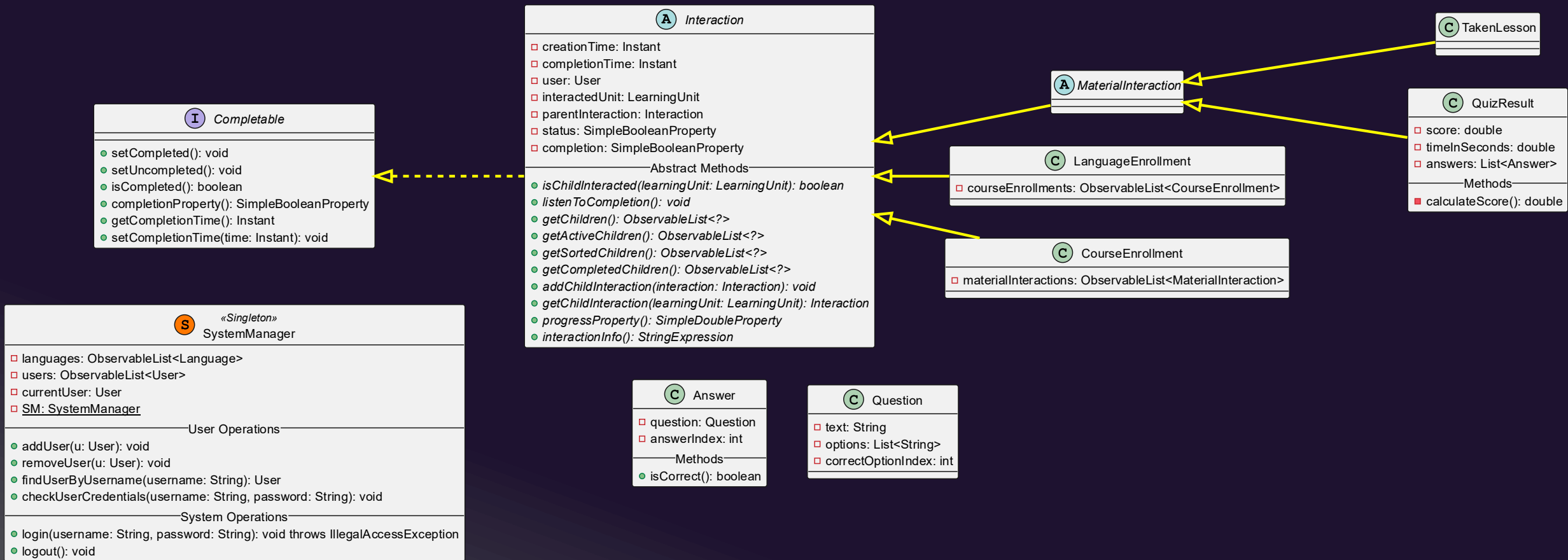
UML Class Diagram

«Interface» Comparable
- compareTo(Object o): int

C Language
- code: SimpleStringProperty
- courses: ObservableList<Course>

A LearningUnit
- creationTime: Instant
- parent: LearningUnit
- title: SimpleStringProperty
- description: SimpleStringProperty
- status: SimpleBooleanProperty
- interactions: ObservableList<Interaction>

List Operations
- addInteraction(interaction: Interaction): void

Abstract Methods
- newInteraction(user: User): Interaction
- addChild(child: LearningUnit): void
- getChildren(): ObservableList<?>
- getActiveChildren(): ObservableList<?>
- getSortedChildren(): ObservableList<?>
- getNextChildren(interaction: Interaction): ObservableList<?>
- subInfo(): StringExpression

C Course
- level: SimpleStringProperty
- index: SimpleIntegerProperty
- materials: ObservableList<Material>

C Lesson
- content: SimpleStringProperty

A Material
- index: SimpleIntegerProperty

C Quiz
- questions: ObservableList<Question>
- passingScore: SimpleDoubleProperty
- allowedTime: SimpleDoubleProperty

Overridden Methods
- addInteraction(interaction: Interaction): void
- subInfo(): StringExpression

I Timestamped
- getCreationTime(): Instant

A User
- creationTime: Instant
- name: SimpleStringProperty
- username: SimpleStringProperty
- password: SimpleStringProperty
- status: SimpleBooleanProperty

Methods
- checkPassword(password: String): boolean

C Student
- languageEnrollments: ObservableList<LanguageEnrollment>

C Teacher

I Activable
- setActive(): void
- setInactive(): void
- isActive(): boolean
- statusProperty(): SimpleBooleanProperty

A Interaction
- creationTime: Instant
- completionTime: Instant
- user: User
- interactedUnit: LearningUnit
- parentInteraction: Interaction
- status: SimpleBooleanProperty
- completion: SimpleBooleanProperty

Abstract Methods
- isChildInteracted(learningUnit: LearningUnit): boolean
- listenToCompletion(): void
- getChildren(): ObservableList<?>
- getActiveChildren(): ObservableList<?>
- getSortedChildren(): ObservableList<?>
- getCompletedChildren(): ObservableList<?>
- addChildInteraction(interaction: Interaction): void
- getChildInteraction(learningUnit: LearningUnit): Interaction
- progressProperty(): SimpleDoubleProperty
- interactionInfo(): StringExpression

A MaterialInteraction

C TakenLesson

C QuizResult
- score: double
- timeInSeconds: double
- answers: List<Answer>

Methods
- calculateScore(): double

C LanguageEnrollment
- courseEnrollments: ObservableList<CourseEnrollment>

C CourseEnrollment
- materialInteractions: ObservableList<MaterialInteraction>

I Completable
- setCompleted(): void
- setUncompleted(): void
- isCompleted(): boolean
- completionProperty(): SimpleBooleanProperty
- getCompletionTime(): Instant
- setCompletionTime(time: Instant): void

S «Singleton» SystemManager
- languages: ObservableList<Language>
- users: ObservableList<User>
- currentUser: User
- SM: SystemManager

User Operations
- addUser(u: User): void
- removeUser(u: User): void
- findUserByUsername(username: String): User
- checkUserCredentials(username: String, password: String): void

System Operations
- login(username: String, password: String): void throws IllegalAccessException
- logout(): void

C Answer
- question: Question
- answerIndex: int

Methods
- isCorrect(): boolean

C Question
- text: String
- options: List<String>
- correctOptionIndex: int

Omar_Mohamed_Hassan

I *Comparable*

● compareTo(Object o): int

C Language

▫ code: SimpleStringProperty
▫ courses: ObservableList<Course>

A *LearningUnit*

▫ creationTime: Instant
▫ parent: LearningUnit
▫ title: SimpleStringProperty
▫ description: SimpleStringProperty
▫ status: SimpleBooleanProperty
▫ interactions: ObservableList<Interaction>

──── List Operations ────
● addInteraction(interaction: Interaction): void
──── Abstract Methods ────
newInteraction(user: User): Interaction
addChild(child: LearningUnit): void
getChildren(): ObservableList<?>
getActiveChildren(): ObservableList<?>
getSortedChildren(): ObservableList<?>
getNextChildren(interaction: Interaction): ObservableList<?>
subInfo(): StringExpression

C Course

▫ level: SimpleStringProperty
▫ index: SimpleIntegerProperty
▫ materials: ObservableList<Material>

C Lesson

▫ content: SimpleStringProperty

A *Material*

▫ index: SimpleIntegerProperty

C Quiz

▫ questions: ObservableList<Question>
▫ passingScore: SimpleDoubleProperty
▫ allowedTime: SimpleDoubleProperty

──── Overridden Methods ────
● addInteraction(interaction: Interaction): void
● subInfo(): StringExpression

I *Timestamped*

● getCreationTime(): Instant

I *Activable*

● setActive(): void
● setInactive(): void
● isActive(): boolean
● statusProperty(): SimpleBooleanProperty

Ammar_Mohamed_Abdelghany

Mohamed_Elsayed_Moustafa

Interaction (A) *abstract*

- creationTime: Instant
- completionTime: Instant
- user: User
- interactedUnit: LearningUnit
- parentInteraction: Interaction
- status: SimpleBooleanProperty
- completion: SimpleBooleanProperty

Abstract Methods

- *isChildInteracted(learningUnit: LearningUnit): boolean*
- *listenToCompletion(): void*
- *getChildren(): ObservableList<?>*
- *getActiveChildren(): ObservableList<?>*
- *getSortedChildren(): ObservableList<?>*
- *getCompletedChildren(): ObservableList<?>*
- *addChildInteraction(interaction: Interaction): void*
- *getChildInteraction(learningUnit: LearningUnit): Interaction*
- *progressProperty(): SimpleDoubleProperty*
- *interactionInfo(): StringExpression*

Completable (I) *interface*

- setCompleted(): void
- setUncompleted(): void
- isCompleted(): boolean
- completionProperty(): SimpleBooleanProperty
- getCompletionTime(): Instant
- setCompletionTime(time: Instant): void

«Singleton» SystemManager (S)

- languages: ObservableList<Language>
- users: ObservableList<User>
- currentUser: User
- SM: SystemManager

User Operations

- addUser(u: User): void
- removeUser(u: User): void
- findUserByUsername(username: String): User
- checkUserCredentials(username: String, password: String): void

System Operations

- login(username: String, password: String): void throws IllegalAccessException
- logout(): void

MaterialInteraction (A) *abstract*

TakenLesson (C)

QuizResult (C)

- score: double
- timeInSeconds: double
- answers: List<Answer>

Methods

- calculateScore(): double

LanguageEnrollment (C)

- courseEnrollments: ObservableList<CourseEnrollment>

CourseEnrollment (C)

- materialInteractions: ObservableList<MaterialInteraction>

Answer (C)

- question: Question
- answerIndex: int

Methods

- isCorrect(): boolean

Question (C)

- text: String
- options: List<String>
- correctOptionIndex: int

Encapsulation

```java
// Setters

public void setTitle(String title) {
    this.title.set(title);
}


public void setDescription(String description) {
    this.description.set(description);
}

// Getters

@Override
public Instant getCreationTime() {
    return creationTime;
}


public LearningUnit getParent() {
    return parent;
}


public SimpleStringProperty titleProperty() {
    return title;
}


public SimpleStringProperty descriptionProperty() {
    return description;
}


@Override
public SimpleBooleanProperty statusProperty() {
    return status;
}


public ObservableList<Interaction> getInteractions() {
    return interactions;
}
```

Ⓐ *LearningUnit*

☐ creationTime: Instant
☐ parent: LearningUnit
☐ title: SimpleStringProperty
☐ description: SimpleStringProperty
☐ status: SimpleBooleanProperty
☐ interactions: ObservableList<Interaction>

Abstract\Interface and Polymorphism
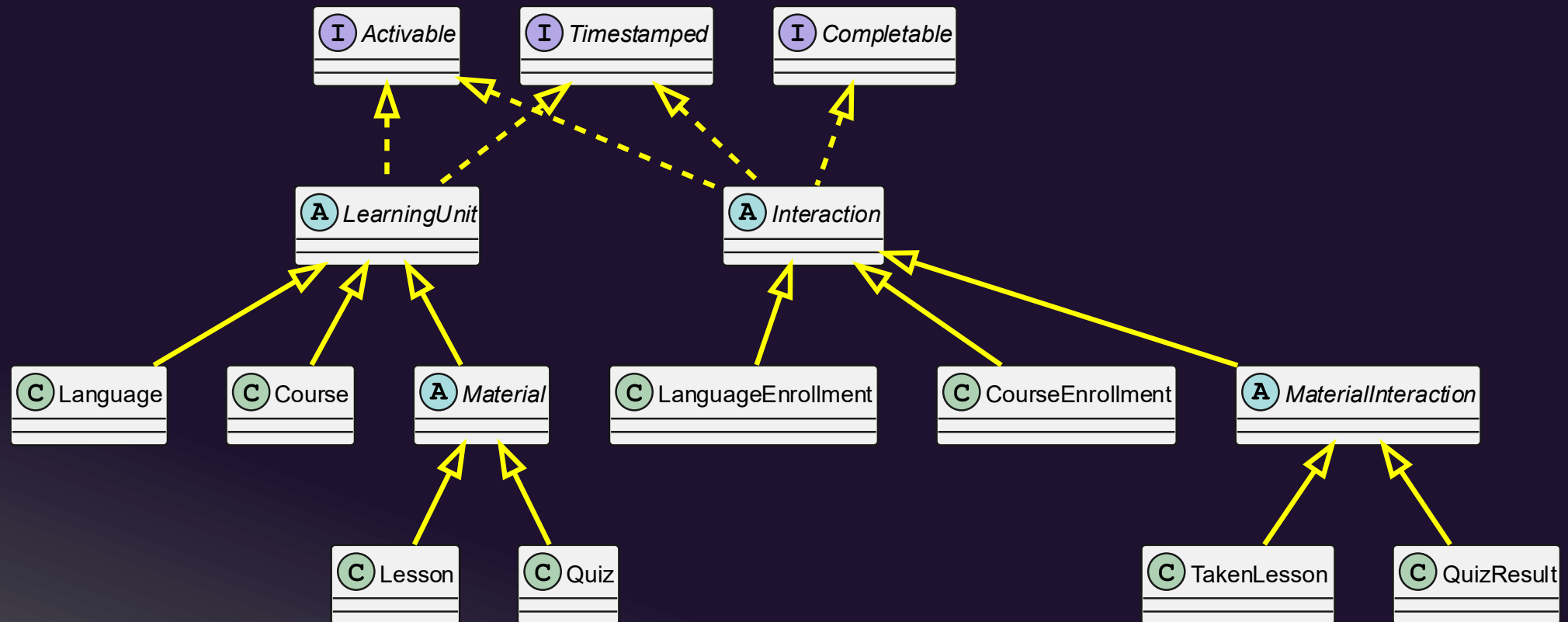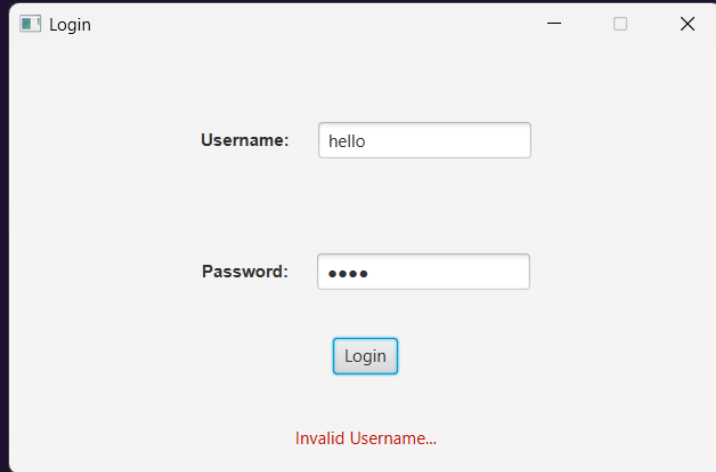
Generic Sort
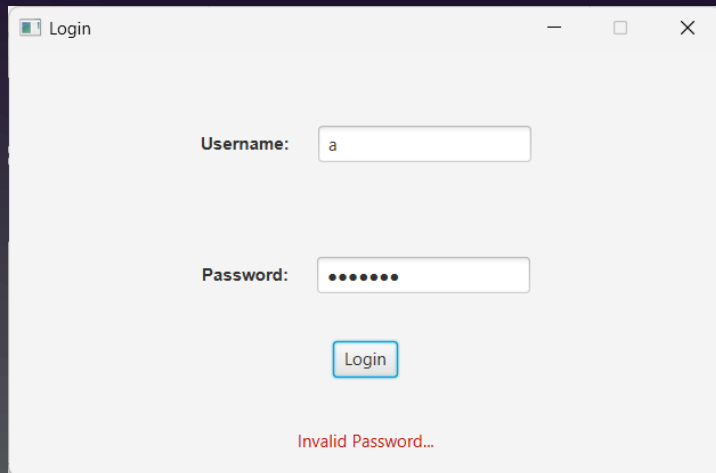
```java
@Override
public int compareTo(Language other) {
    return titleProperty().get()
        .compareTo(other.titleProperty().get());
}
```
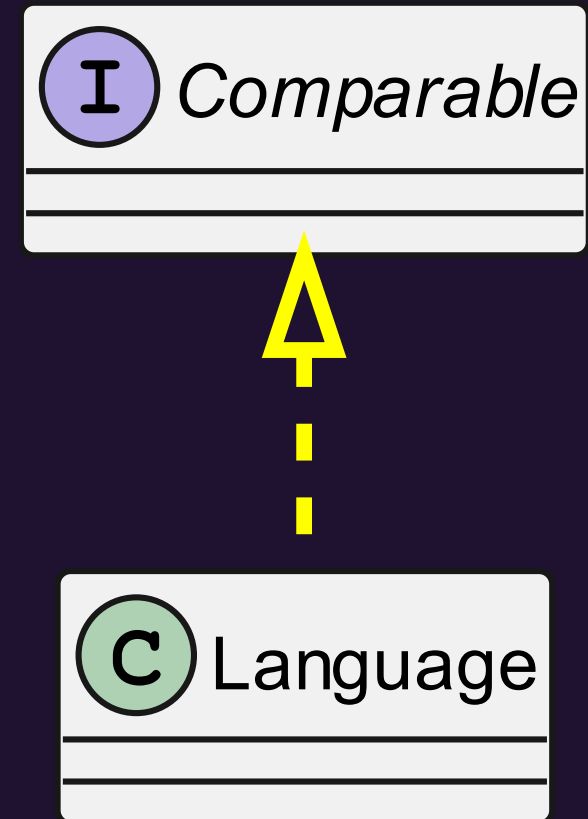
Exception Handling


```java
// In SystemManager

public void checkUserCredentials(String username, String password) {
    User user = findUserByUsername(username);
    if (user == null) {
        throw new IllegalArgumentException("Invalid Username...");
    }
    if (!user.checkPassword(password)) {
        throw new IllegalArgumentException("Invalid Password...");
    }
}


public void login(String username, String password) throws IllegalAccessException {
    User user = findUserByUsername(username);
    if (user == null || !user.checkPassword(password)) {
        throw new IllegalAccessException("Login Failed...");
    } else {
        currentUser = user;
    }
}


// In LoginStage

@FXML
void login(ActionEvent event) {
    try {
        // ...
        SM.login(tfUsername.getText(), pfPassword.getText());
        // ...
    } catch (Exception ex) {
        lbError.setText(ex.getMessage());
    }
}
```

GUI

```java
public class StudentScene extends Scene {

    @FXML
    private Button btEnrollNewLanguage;

    @FXML
    private HBox hBox;

    @FXML
    private Label lbWelcome;

    @FXML
    private AnchorPane root;

    @FXML
    void logout(ActionEvent event) {

    }

    @FXML
    void openEditProfile(ActionEvent event) {

    }

    @FXML
    void showAvailableLanguages(ActionEvent event) {

    }

}
```
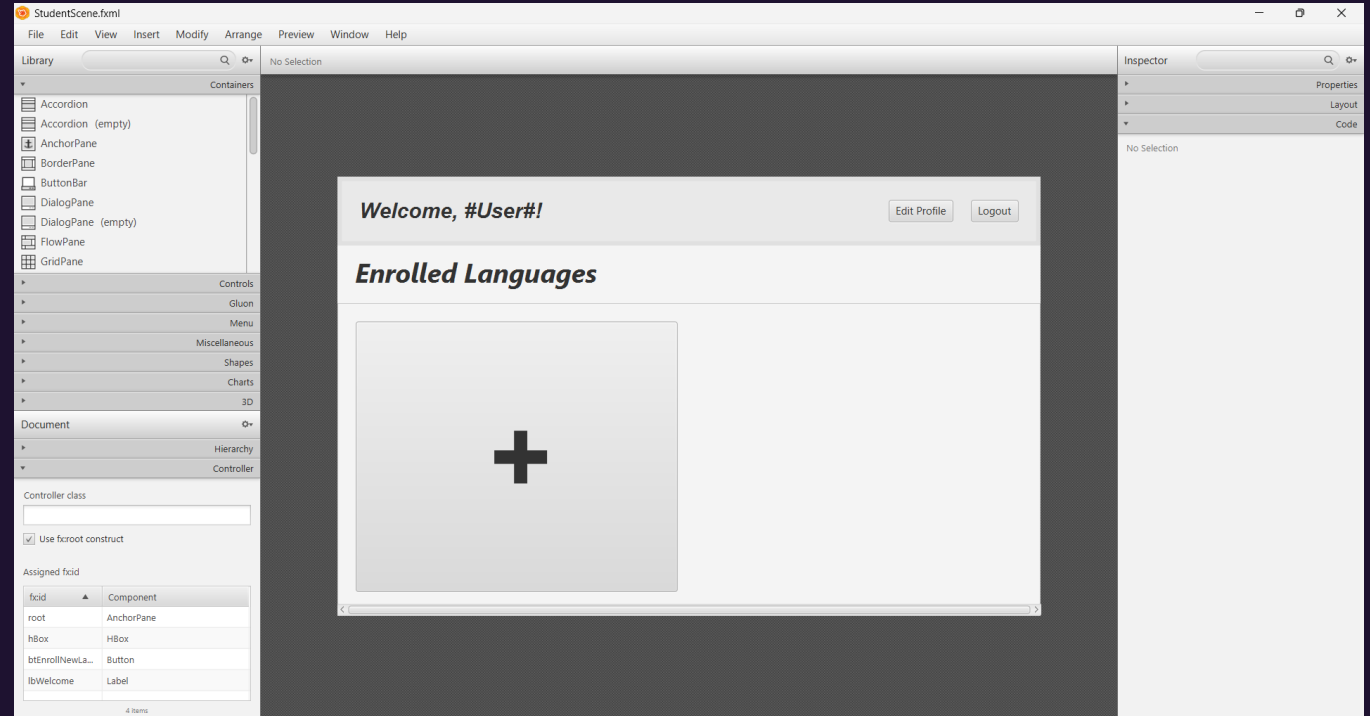
```java
System.out.println("Coding is like a joke, when you explain it, it's bad!");

System.out.println("Thank You!");
```